Lab Manual: 03

Lab Topic: Loops and Array Manipulations

Course Code: CSE1116 (Object Oriented Programming Laboratory)

Course Instructor: Mir Moynuddin Ahmed Shibly, Lecturer, CSE, UIU.

Lab Objective

1. **Apply** concepts of loops, arrays, and ArrayLists.
2. **Write** and **execute** programs using these concepts in Java.

Lab Activities

A. Arrays in Java

- Arrays are declared in the following manner in Java.

```
int[] numbers = new int[10];
```

- The above-mentioned line created an integer array of numbers of size 10.
- The array elements can be accessed in the same way as in C.
- Examine the following program.

```
import java.lang.*;
import java.util.*;
class SampleArray{
 public static void main(String[] args){
    // initializing an integer array numbers with size 10
    int[] numbers = new int [10];
    Scanner input = new Scanner (System.in);
    // assigning values randomly
    for(int i=0; i<numbers.length; i++){
          numbers[i] = input.nextInt();

          // numbers[i] = (int)(Math.random()*100);// Read random numbers
    }
    // displaying the values
    for(int i=0; i<numbers.length; i++){
          System.out.print(numbers[i] + " ");
    }
 }
}
```

- **numbers.length** returns the size of the array.
- For Java String, you can use **charAt(i)** method to access the character at i^{th} position of the string. Suppose,

```
String str = "Hello World";
System.out.print(str.charAt(0)); // returns the character 'H'
```

B. Java ArrayList

Sometimes it's better to use dynamic size arrays. Java's Arraylist can provide you this feature. Try to solve this problem using Arraylist.

```
ArrayList<ArrayList> listArray = new ArrayList<ArrayList>();****
 ArrayList< Integer > intArrayList =  new ArrayList< Integer >();
 for(int j=0;j<numOfIntegers;j++){
      intArrayList.add(new Integer(sc.nextInt()));
 }
 listArray.add(intArrayList);
```

C. Enhanced For loop

- Java also includes another version of for loop
- Enhanced for loop provides a simpler way to iterate through the elements of a collection or array
- It is read only loop where you can't update the values as opposite to other loops

```
public class enhancedforloop
{
    public static void main(String args[])
    {
        int array[] = {10, 20, 30)

        //enhanced for loop
        for (int x:array)
        {
            System.out.println(x);
        }

        /* for loop for same function
        for (int i = 0; i < array.length; i++)
        {
            System.out.println(array[i]);
        }
        */
    }
}
```

Lab Problems

01: Write a Java program that reads ten integers from the console and determines how many positive and negative values have been read and computes the total and average of the input values (not counting zeros). Your program ends with the input **0**. Display the average as a floating-point number.

02: Write a Java program that prompts the user to enter two positive integers and displays the GCD. The greatest common divisor (GCD) of two integers' **n1** and **n2** can be found in the following way: First find **d** to be the minimum of **n1** and **n2**, and then check whether **d**, **d-1**, **d-2**, **2**, or **1** is a divisor for both **n1** and **n2** in this order. The first such common divisor is the greatest common divisor for **n1** and **n2**. *Use appropriate JOptionPane dialog boxes to read inputs and write outputs*.

03: Write a Java program that reads student scores, gets the best score, and then assigns grades based on the following scheme:

Grade is A if score is ≥ best - 10
Grade is B if score is ≥ best - 20;
Grade is C if score is ≥ best - 30;
Grade is D if score is ≥ best - 40;
Grade is F otherwise.

The program prompts the user to enter the total number of students, then prompts the user to enter all of the scores and concludes by displaying the grades. Here is a sample run:

```
Enter the number of students: 4 Enter
Enter 4 scores: 40 55 70 58 Enter
Student 0 score is 40 and grade is C
Student 1 score is 55 and grade is B
Student 2 score is 70 and grade is A
Student 3 score is 58 and grade is B
```

04: Write a Java program that reads the integers between 1 and 100 and counts the occurrences of each. Assume the input ends with **0**. Here is a sample run of the program:

```
Enter the integers between 1 and 100: 2 5 6 5 4 3 23 43 2 0 ↵Enter
2 occurs 2 times
3 occurs 1 time
4 occurs 1 time
5 occurs 2 times
6 occurs 1 time
23 occurs 1 time
43 occurs 1 time
```

05: A *Palindrome* is a number or a string that reads the same from either way (forward or backward). As an example, RADAR is a palindrome, but ROVER is not. Write a Java program that prompts the user to enter a string and displays whether the string is a palindrome or not. You may not use any built-in Java methods to accomplish that.

06: Write a Java program that reads in ten numbers and displays the number of distinct numbers and the distinct numbers separated by exactly one space (i.e., if a number appears multiple times, it is displayed only once). (*Hint*: Read a number and store it in an array if it is new. If the number is already in the array, ignore it.) After the input, the array contains the distinct numbers. Write a method Isdistinct() to check whether a number exists multiple times or not.

07: Write a Java program that randomly generates an integer array/ArrayList, *numbers*, of size 100. Then, find the value and index (position) of the second highest and the second smallest element. Use separate methods for determining the highest and smallest element.

08: Write a program that prompts the user to enter the number of students, the students' names, and their scores, and prints student names in decreasing order of their scores.

09: You are given n lines. In each line there are zero or more integers. You need to answer a few queries where you need to tell the number located in y^{th} position of x^{th} line. Take your input from System.in.

Input Format
The first line has an integer n. In each of the next n lines there will be an integer d denoting number of integers on that line and then there will be d space-separated integers. In the next line there will be an integer q denoting number of queries. Each query will consist of two integers x and y.

Output Format
In each line, output the number located in y^{th} position of $x^{th\ line}$. If there is no such position, just print "ERROR!"

Sample Input

```
5
5 41 77 74 22 44
1 12
4 37 34 36 52
0
3 20 22 33
5
1 3
3 4
3 1
4 3
5 5
```

Sample Output

```
74
52
37
ERROR!
ERROR!
```

10: Write a function that will return an array/ArrayList consisting of the first n term of the Fibonacci sequence.

Example 1:
Input: 5
Output: [0, 1, 1, 2, 3]

Example 2:
Input: 8
Output: [0, 1, 1, 2, 3, 5, 8, 13]

11: Write a program to find the largest word in a sentence.

Example 1:
Input: "Welcome to Java Programming"
Output: "Programming"

Example 2:
Input: "Computer Science and Engineering is easy"
Output: "Engineering"

12: Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.

Example 1:
Input: nums = [1,2,3,1]
Output: true

Example 2:
Input: nums = [1,2,3,4]
Output: false

Example 3:
Input: nums = [1,1,1,3,3,4,3,2,4,2]
Output: true

13: Given an array of integers and an integer target, return *indices of the two numbers such that they add up to the target*. You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

Example 1:
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:
Input: nums = [3,2,4], target = 6
Output: [1,2]

Example 3:
Input: nums = [3,3], target = 6
Output: [0,1]

14: Given an integer array, move all 0's to the end of it while maintaining the relative order of the non-zero elements. **Note** that you must do this in-place without making a copy of the array.

Example 1:
Input: nums = [0,1,0,3,12]
Output: [1,3,12,0,0]

Example 2:
Input: nums = [0]
Output: [0]

15: Given an integer array, rotate the array to the right by k steps, where k is non-negative.

Example 1:
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]

Example 2:
Input: nums = [-1,-100,3,99], k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]

16: Write a Java program to delete all the duplicates from an array/ArrayList. Assume the elements of the array will not be larger than 100.

Example 1:
Input: nums = [1,2,3,1]
Output: [1,2,3]

Example 2:
Input: nums = [1,2,3,4]
Output: [1,2,3,4]

Example 3:
Input: nums = [1,1,1,3,3,4,3,2,4,2]
Output: [1,3,4,2]